EXPERIENCES WITH A SURVEY TOOL FOR DISCOVERING NETWORK TIME PROTOCOL SERVERS

James D. Guyton & Michael F. Schwartz

CU-CS-704-94 January 1994

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JAN 1994**	2. REPORT TYPE	3. DATES COVERED **00-00-1994 to 00-00-1994**

4. TITLE AND SUBTITLE
Experiences with a Survey Tool for Discovering Network Time Protocol Servers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Computer Science,University of Colorado,Boulder,CO,80309

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**12**	

EXOERIENCES WITH A SURVEY TOOL FOR DISCOVERING NETWORK TIME PROTOCOL SERVERS

CU-CS-704-94 1994

James D. Guyton & Michael F. Schwartz

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430 USA

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR(S) AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE ACKNOWLEDGMENTS SECTION.

Experiences with a Survey Tool for Discovering Network Time Protocol Servers

James D. Guyton, Michael F. Schwartz – University of Colorado, Boulder

Technical Report CU-CS-704-94
January, 1994

ABSTRACT

The Network Time Protocol (NTP) is widely used to synchronize computer clocks throughout the Internet. Existing NTP clients and servers form a very large distributed system, and yet the tools available to observe and manage this system are fairly primitive. This paper describes our experiences with a prototype tool that attempts to discover relevant information about every NTP site on the Internet. The data produced by this tool can be used for a variety of purposes, including locating nearby accurate time servers and computing aggregate and long-term evaluations of the size and health of the NTP system. Importantly, our tool provides a means by which new NTP server administrators can make informed choices among the possible servers with which to synchronize, balancing the need for accurate time with the need to distribute server load. This is an important step towards improving global NTP system scalability, since at present our measurements indicate that the high-stratum servers are heavily overloaded.

Introduction

Clock Synchronization and NTP

Clock synchronization is useful for a wide variety of purposes, particularly in a network environment. Uses include keeping accurate file timestamps in distributed filesystems (e.g. so that *make* doesn't get confused), ticket validation timers in security systems like Kerberos [SNS88], and potentially even synchronizing clocks across the globe for very-long baseline radio astronomy work.

Designed and developed by David Mills of the University of Delaware, the Network Time Protocol (NTP) [MIL92] is currently used by thousands of Internet hosts to synchronize their local clocks to within a few milliseconds of the international time standard. Systems to distribute accurate time have been studied for some time [LIN80, BRA80], but NTP's contribution is that it is able to distribute accurate time over the unpredictable Internet. Hosts participating in time distribution via NTP form a hierarchical, master-slave, self-organizing subnet[1] of the Internet. At the top layer of the hierarchy are the sources of very accurate time. These *stratum-1* servers usually have local atomic clocks (that are synchronized by means other than NTP) or radio receivers that decode time signals broadcast by national standards organizations (e.g. WWV timecodes broadcast by NIST in Ft. Collins, Colorado).

NTP goes to great lengths to distinguish time servers with accurate data (*truechimers*) from those with false data (*falsetickers*), and to obtain the best synchronization possible in the face of widely varying Internet delays. Recent versions of NTP include algorithms to combine the offsets of several clocks, to construct a synthetic time more accurate than any individual time server.

In addition to its value for supporting clock synchronization, running an NTP server can indirectly help network managers uncover and diagnose network problems, based on the statistics NTP maintains. For example, NTP keeps a status register of how "reachable" its servers have been recently. This information provides a crude but useful measure of packet loss, which can be helpful in diagnosing network load or connectivity instabilities.

We chose to study NTP because it is an important and widely distributed system. At present, it is "bundled" with the software distributions from a number of workstation manufacturers, and a number of large organizations use NTP [MIL93b]. The U.S. Weather Service uses NTP, and soon every Public Broadcasting

[1] The word "subnet" is used in this paper to refer to the subset of Internet hosts that use the NTP protocol. It should not be confused with the more common usage, which denotes partitioning a single IP network number into multiple smaller networks.

Service station affiliate will be a client. Merrill Lynch is currently populating its worldwide network with NTP.

NTP Management Problems

While NTP goes to great lengths to maintain well-synchronized clocks in the face of unpredictable Internet behavior, at present managing the NTP network itself is quite difficult. The latest NTP software distribution includes a few debugging tools for examining and changing the state of an individual server, but does not include tools to discover the nearest NTP server or to help debug the NTP system as a whole. When only a few sites ran NTP, this information was easy to gather by manual methods. But now that there are many thousands of sites running some version of NTP, the need for additional discovery and management tools has become painfully clear.

The largest problem is that the stratum-1 servers are seriously overworked and in danger of becoming saturated. In principle this should not be a problem, because NTP allows time to be distributed hierarchically. The problem arises because it is not trivial for an NTP server administrator to pick an appropriate set of servers with which to "chime" (i.e., synchronize clocks) when first joining the NTP server network. As a result, far too many administrators select high-level NTP servers (see Table 3). An NTP discovery system that allowed new administrators to identify "nearby" servers would reduce this problem, and markedly enhance global system scalability. Running periodic surveys would also make it possible for regional network administrators to monitor the configuration of the NTP subnet within their domain and apply social pressure to fix poorly configured NTP hosts.

An alternative approach would be to restrict which systems could chime with high-stratum NTP servers, through an access control mechanism. While the code for this approach has been implemented, it has not been widely adopted. In our opinion this approach should be avoided – it would result in more work for the administrators of the restrictive servers, while merely increasing the load on the remaining unrestricted servers. Because the problem arises from the inability to obtain good information, a solution based on discovery seems more appropriate than one based on access control.

In addition to supporting more well-informed configuration management choices, an NTP discovery/survey tool is useful for helping to debug and understand the NTP protocol itself. To our knowledge there has never been an aggregate picture of the "state of the NTP network" at a finer level of detail than a simple count of the number of hosts running NTP. We hope that by offering the ability to collect meaningful measurements of the state of the NTP world, a deeper understanding of the workings of this large, distributed system can be attained, which would enable further improvements to the NTP model.

Survey Methodology

There is no complete registry of hosts that run NTP. Instead, we begin with a list of hosts known to run NTP, query each host with an NTP information request, parse the response, and add any previously unseen hosts to the database. These newly discovered hosts are then queried and the process repeats until no new hosts are discovered.

At first examination, one might believe that this iterative survey process would quickly discover all Internet NTP hosts. For a variety of reasons, this is not so. In the following sections we discuss the problems and the prototype's approach to solving them.

Monitor List Queries

The biggest difficulty in implementing an NTP survey is that the protocol does not require each NTP time server to keep track of its clients. While all NTP implementations require clients to track the server with which they chime, they may keep little or no information about clients that chime with them. This makes it easy to move up the NTP hierarchy, but often impossible to move down. Given that we know the most about the top two levels of the tree and relatively little about the leaves, this is a serious restriction.

While keeping track of clients is not required by the NTP specification, it is frequently very useful for debugging. Client tracking is supported by the XNTP implementation of NTP Version 3 [XNTP93]. This debugging feature of XNTP is called "monitor mode", and the results of monitoring can be retrieved remotely with the "monitor list" command.

System and Peer Variable List Queries

Versions 2 and 3 of the NTP protocol specify a standard control packet format that allows internal NTP variables to be examined and set. These variables contain useful information about the state of the local NTP system, the software phase-locked-loops and filters that it uses, the status of the peers with which a host is chiming, and a wealth of other information. These queries provide a good source of information for an NTP survey.

An immediate problem with an NTP survey tool is to choose what variables should be requested. The current solution to this problem is to allow the user to specify the set of interesting variables. A configuration file lists all of the system and peer variables to be used in queries, and any returned values for these variables are recorded by the survey.

One problem with variable queries is that different implementations of NTP can have slightly different spellings of some of the variable names, resulting in query failures when a server is asked for non-existent variables. What makes this minor problem much worse is the fact that, while a variable query command can contain an long list of variable names whose values are to be returned, all observed NTP implementations simply return an error if *any* of the variable names are unknown to that implementation. This led to the need for a rather complex query/error/retry algorithm to extract data from each NTP server.

A smaller problem with variable queries was caused by the fact that query responses are returned formatted for human-consumption, including white space, punctuation and newlines. While this response makes it easy to write simple query/display programs, it made implementation somewhat more difficult for our NTP survey tool.

Version Issues

The NTP protocol is an evolving entity, and various implementors have made substantial improvements with four major versions of the protocol over the past eight years. But as with any widely deployed system, there have been a few compromises made to facilitate backwards compatibility.

The basic idea is that newer versions of NTP may interoperate with systems running older versions, but when they do so it is suggested that they "fib" about their own version number so as not to confuse the remote system. This is wonderful from a compatibility point-of-view, but makes it difficult for our survey tool to determine correct version numbers.

Access Restrictions

A potential problem that we ignored in our current prototype is the issue of authentication. Given that a malicious user could try and skew clocks by supplying faulty NTP times, the NTP protocol specification includes support to authenticate requests. Unfortunately, an NTP server that uses authentication is not queryable by our discovery tool. Even though the authentication code is widely deployed, the need for it has not become severe, and at least for the moment an NTP explorer can blissfully explore the network without worrying about lack of permissions and magic keys.

One reason NTP-based access controls are not often used is that sites often use firewall gateways [CQ92] to control all incoming traffic, rather than setting up restrictions on a per-service basis. These gateways present difficulties for our survey tool because there is no easy method of determining whether a potential NTP host is behind a firewall. From the survey tool's perspective the host appears to be "temporarily" unreachable. The only solution we can see for this problem is to add survey functions to the firewalls so that (approved) statistics of what's going on behind the firewall can be exported to survey tools like this one.

One final access restriction note concerns a lesson learned by other network information discovery projects: Some Internet system administrators consider network discovery methods to be distressingly equivalent to trespassing. Fortunately, this is such a small percentage of the Internet community that their hosts can safely be omitted from the survey without seriously affecting the survey results. The prototype NTP explorer module has a simple method of host and network avoidance. We initialized its "don't trespass" database from a list of systems whose administrators had previously requested to be left alone.

Implementation

The current implementation consists of two query programs written in C, and a small collection of utilities and filters written in C and PERL. The two query modules both take a list of IP addresses as input, do either a monitor-list or variable query, and update the databases with their results.

There are two very different types of data that need to be collected by the NTP survey tool. One type of data is the NTP topology and the set of hosts that participate in the NTP subnet. These data are relatively small and easy to manage.

The other type of data that need to be collected by an NTP survey is a large amount of NTP state information giving the status of the NTP protocol engine at each reachable NTP host in the Internet. These data need to be recorded and made available to analysis tools to understand how the NTP network is performing and changing over time.

The current implementation splits these two types of data into different databases. One is a standard UNIX "dbm" file that contains minimal information about each host that is an actual or suspected participant in the NTP subnet, along with the timestamps and status codes of recent NTP query attempts. The other is a relational database recording all the detailed NTP data that are gathered by a particular survey run. These later data are stored in an RDB [HOB93] database, which supports a simple relational model that eases manipulation and analysis.

We seed the survey database from a list of publicly available stratum-1 and stratum-2 time servers manually maintained by Mills.[2] At present this file lists approximately 35 stratum-1 servers and 70 stratum-2 servers. Since this file is not designed to be parsed electronically, we manually extracted time server host names from this file, and used them to seed the survey database. This initialization program is the only module in the prototype that allows the use of full domain-style hostnames. The rest of the implementation simply uses IP addresses in the interest of performance. Additional hosts to check can be added to the database by hand or by other means (e.g. a Fremont explorer module that has reason to believe that a host may be chiming NTP; see the Related Work section for a discussion of Fremont).

Survey Experiences and Results

The monitor list query uses a packet format that the NTP Version 3 standard defines only as "reserved for private use." Fortunately, the monitor list command is often available, because it is included in the widely deployed XNTP implementation. However, not all sites that run NTP use XNTP, and many that do run XNTP leave monitor mode disabled (it is disabled by default). Moreover, as mentioned earlier, some of those that collect monitor data require prior authorization to use the "monitor list" query and retrieve the information.

Even with this daunting list of restrictions, it turns out that there are enough publicly retrievable monitor listings that using this style of query resulted in gathering evidence of approximately 10,000 possible NTP hosts in about 8 hours of survey time.

By comparison, the survey module that queries for system and peer variables plods along gathering a great deal of information, but adds comparatively few hosts to the database of potential NTP systems. It took about 50 hours to attempt to query 10,000 hosts.[3]

The number of NTP hosts found in the initial survey was relatively large. The main database contains over 15,000 unique IP addresses of hosts that we have reason to believe speak NTP, and the survey was able to speak NTP directly with over 7,200 systems. While this database contains no duplicate IP addresses, hosts with multiple network interfaces may be counted twice.

In comparison with other NTP surveys, our survey has done rather well. Mill's survey of July 1993 [MIL93a] found a total of 6,185 hosts (via monitor list), while a survey done by Pruy in October 1993 [PRUY93] was able to communicate with about 2,100 hosts.

The results of an NTP survey are a little tricky to summarize without being misleading. The set of NTP hosts in the database built by recursively running the "monitor list" command is much larger than the set of NTP hosts that are actually reachable by NTP information queries. Therefore, we distinguish the results below by data source.

Statistics from "Monitor List" Queries

The monitor facility of XNTP records information about a particular server's clients. The IP addresses and NTP protocol version number of its clients are recorded, but very little else is recorded. As described before, the version number can be fictitious, and must be taken with a dose of skepticism.

The total number of IP hosts derived from 1,760 hosts that responded to monitor lists queries are listed in Table 1. It is interesting that, although the majority of sites are running the most recent version of NTP, many sites still have not upgraded. Part of the problem is that a number of workstation manufacturers are bundling outdated versions of NTP [MIL93b]. It would be interesting to collect these measurements periodically as the NTP subnet continues to grow, and see what percentage of "old" hosts upgrade vs. how many new hosts start by running the latest version.

Version 3	7,615
Version 2	2,432
Version 1	2,095
Version 0	58
Total Hosts	12,200

Table 1: NTP Host Count

Statistics from "Variable List" Queries

The data from the peer and system variable queries are more accurate than those from monitor list queries, though not nearly as complete. Even so, an amazing amount of raw data from our survey is available for analysis. The following statistics are but a first-pass at mining

[2] Available by anonymous FTP from louie.udel.edu, /pub/ntp/doc/clock.txt

[3] The long survey times are a result of the current implementation's use of sequential reads and timeouts.

interesting information from it.

While there were over 15,000 hosts in the completed survey's database of suspected NTP hosts, only 7,251 responded to NTP variable list queries. The statistics in Table 2 are summarized from the data returned by these 7,251 hosts. Clearly, the high-level strata are overused. At present, Mills attempts to reduce this problem periodically by sending a message on the NTP mailing list asking people to back off of the stratum-1 servers and to make more use of the stratum-2 servers. Perhaps if our NTP discovery tool were built into the system (so that new site administrators could choose a good peer with which to chime), this would be less of a problem.

Note that stratum 16 is defined by the NTP protocol specification as infinitely far away from the time source. The hosts that claim to be at strata 13 through 15 have more subtle problems. At first examination they appear to be a collection of isolated hosts in Germany with their own time source that believes itself to be at stratum-13, plus an extremely confused set of hosts at the University of Tennessee.

Stratum	# Hosts
1	66
2	1,476
3	3,374
4	2,001
5	38
6	6
7	1
8-12	0
13	2
14	32
15	1
16	254

Table 2: NTP Hosts per Stratum

The average number of clients per server can be directly computed from the above table, and are shown in Table 3. This table clearly indicates how poorly the stratum-2 (and lower) servers are utilized.

Central to the NTP clock calculations are the delay and dispersion values between a client and its server. *Delay* is the round-trip network delay between the peer and its server. *Dispersion* is the computed maximum error of the peer clock relative to its server. Both quantities are typically measured in milliseconds. Table 4 shows some statistics regarding the aggregate delay and dispersion between clients and their servers, grouped by the stratum level of the servers.

Server Stratum	Clients Per Server
1	22.36
2	2.29
3	0.59
4	0.02
5	0.16
6	0.17

Table 3: NTP Tree Branching

Rootdelay is the estimated total delay to the top of the NTP tree, while rootdispersion provides an error bound for how far off the local clock is from the NTP stratum-1 time source. This is probably as good a metric to estimate the health of the NTP subnet as any. Table 5 summarizes the rootdelays and rootdispersions from clients at each stratum to their NTP time source.

Tables 4 and 5 show that there is a fairly small median error as the time is distributed from stratum to stratum, with the average being substantially higher. In other words, many low-stratum servers offer very good time, but some offer very bad time.

Without advance knowledge of "how good" a set of time servers might be, people tend to pick high-stratum servers, because servers at the top of the distribution tree would intuitively seem to provide more accurate times. Yet, as Tables 4 and 5 show, this is not necessarily the case. In fact, because of the new NTP clock-synthesis code, it is actually possible that a low-stratum clock may be more accurate than any of its parents in the distribution tree. Tools are needed so that new NTP server administrators can make informed choices among the possible servers with which to chime, so that they can pick a server that provides accurate time without overloading the high-level servers.

Observations

After examining the results of our survey, a number of unexpected results emerged. The most surprising are enumerated below:

- A large number of hosts run NTP version 1. Out of the 15,000 hosts in the database, the NTP survey tools were able to speak NTP with about 7,200 hosts. About 7,700 of the remaining hosts did not respond to either Version 2 or Version 3 NTP packets. A small test program revealed that 5,600 of these hosts were reachable[4], and of these 2,350 hosts would respond to an NTP version 1 date query.

[4] The program simply tested whether it could get a response from the host's UDP echo port.

- Many hosts appear in monitor lists that do not run NTP servers. The numbers mentioned above imply about 3,300 hosts that are reachable and are suspected of running NTP (i.e., appear in the database for one reason or another), do not run a full NTP peer. The most likely conclusion is that these hosts use NTP to set their clocks when booting, but do not run the normal NTP server process.

- After sending a query to an NTP host, the response would often come back from an IP address different than what was expected. The NTP code currently treats this as an error, but it should instead be treated as a serendipitous discovery of multiple network interfaces on a gateway.
- While the initial implementation had a simple filter to delete the "loopback" host and other obviously invalid IP addresses before adding them to the database, about 85 of the 15,000 "hosts" still turned out to be network numbers. It would be useful to track down how these network numbers were introduced.

The Need for Stability Measures

One final observation is that the current NTP tools provide a way to observe the time errors *at the time of the query*. However, because NTP periodically resynchronizes clocks, these measures may not capture instabilities. For example, there was a problem with the latest experimental version of NTP when run on HP workstations, that occasionally caused the clock to be incorrect by about 40 years. If a downstream client synchronized with such a clock after this error arose, it could lead to many problems. A global NTP survey/management system should provide measures of clock error that would allow potential clients to avoid such instabilities.

What was Missed?

What percentage of the total do these numbers reflect? It's impossible to know for sure, since there are many firewalls on the Internet hiding an unknown number of NTP hosts. Dave Mills has estimated that there are approximately 100,000 hosts running NTP, tens of thousands of which are behind firewalls [MIL93b]. The only way to know for sure is to get the cooperation of the firewall sites to allow surveys into their domains. In the mean time modifying XNTP to enable the monitor-list feature by default would quickly extend the reach of this survey tool.

Related Work

There are a variety of systems related to the current work. Census is a tool that recursively descends the Domain Naming System (DNS) tree, gathering information from as many sites as possible [GAN92]. Robodoc is a tool that tests remote DNS servers for various configuration errors [MOC93].

Archie gathers directory listing information from "anonymous FTP" servers around the Internet, for use as an indexing/search service [ED92].

The Simple Network Management Protocol (SNMP) defines a general method to query and control network servers [SNMP90]. If the NTP system were being written today, it would probably use SNMP instead of its native

Clients to Stratum	Delay Average	Delay Median	Delay Std. Dev.	Dispersion Average	Dispersion Median	Dispersion Std. Dev.
1	105	79	111	38	20	87
2	42	30	74	46	18	164
3	36	39	62	55	21	125
4	42	43	19	114	73	153
5	50	53	19	112	91	83

Table 4: Client-to-Server Metrics

Clients at Stratum	Root Delay Average	Root Delay Median	Root Delay Std. Dev.	Root Dispersion Average	Root Dispersion Median	Root Dispersion Std. Dev.
2	155	95	177	102	52	213
3	160	104	166	175	114	251
4	184	105	163	195	145	259
5	116	28	182	362	166	419

Table 5: Client-to-Root Metrics

query/control protocol. Perhaps in time NTP will be changed to use SNMP, at which point the NTP survey code can begin to use more standard network management tools.

The Fremont system gathers and cross-correlates data from a number of network protocols and information sources to construct a picture of key network characteristics, such as hosts, gateways, and topology [WCS93]. Each source is collected by a distinct "explorer module", which deposits the gathered data in a network accessible database managed by a *journal server*. The invocation of explorer modules is under the control of a *discovery manager*, which decides what information needs to be collected and which explorer modules should be invoked to collect the data.

Fremont's support for multiple explorer modules, a discovery manager, and journal server, makes the opportunity for synergistic results obvious. The data mined by the current NTP survey tool should eventually be turned into data for the Fremont system.

Conclusions

In this paper we presented a survey tool for discovering the topology of the global NTP server network, and a set of experimental results from running this tool on the Internet. The basic survey methodology involves iteratively expanding a seed list of NTP hosts, based on information retrieved by NTP queries. This approach is complicated by a number of operational issues in the NTP network, ranging from version mismatches to firewall gateways and other limitations.

Our survey provides measurements of the current size and configuration of the NTP server network, uncovering about 10% of the total number of estimated hosts running NTP. The survey was limited primarily by the presence of firewall gateways in the Internet. The results showed that the primary time servers are overloaded, and that the global NTP time distribution tree is very poorly balanced.

To allow continued growth of the NTP community, a more balanced time distribution tree is necessary. Our discovery tool is a first step in developing the tools needed to help extend the scalability of the NTP system. In particular, Tables 4 and 5 demonstrate that the times from low-stratum servers often provide accurate times. Our tool provides a means by which new NTP server administrators can make informed choices among the possible servers with which to chime.

While the total extent of the NTP subnet probably won't be known until more hosts enable the "monitor list" command, the usefulness of this tool may encourage more people to do so. In the meantime, the current sample size seems large enough to collect more than enough information to make it interesting – and hopefully useful – to a wide variety of users.

One might observe that a discovery tool would not be necessary if NTP itself kept records of the server network topology (e.g., maintaining both upwards and downwards pointers, as is done in the DNS tree). There are at least two general reasons NTP does not provide adequate support in this regard. First, when developing any complex algorithm, it behooves software designers to focus on the problem at hand. In retrospect it is usually easy to point out what "should" have been planned into the system from the start, yet it makes sense to postpone worrying about issues that will only arise once a system becomes "wildly successful". Second, it often is not clear from the start what state information should be collected to aid in system management.

From this perspective, our discovery and survey tool provides both a "transition path" towards a future management-oriented version of NTP, and a set of experiences concerning what is needed. Below we discuss planned future work.

Future Work

Performance Improvements

Even though the number of packets sent to do the NTP queries is fairly small,[5] our tool currently takes a long time to complete a survey. The tool could be sped up significantly by using a simple "pipelined" design that allowed for overlapping I/O with timeouts. The major complications are parallel updates to the databases, and the fact that the NTP protocol is based on UDP (and hence the operating system interface is full of dangers of dropping packets). Neither of these are serious obstacles, and a fully parallel version of this explorer should be implemented.

Additional Sources of Potential NTP Hosts

Our survey's successfulness depends critically on what hosts were initially seeded in the survey database. To improve the survey's thoroughness, there should be other methods of seeding this database. Some possible methods include Ethernet monitoring logs, queries of selected hosts found in the DNS system[6], hostnames from messages posted to the netnews group that discusses NTP, hosts that contain "ntp" in their name discovered from DNS traversals, and maybe even checking hosts that have retrieved the NTP source using FTP from

[5] About 5-10 packets per host.

louie.udel.edu.

Configuration Tools

An easily used tool should be written that combines data from an NTP survey with information from routing tables and/or traceroute, and generates a list of suggested time servers for a specific NTP client. An extension of this tool could be written that would accept a set of network numbers (e.g. all the networks in a regional network), and display NTP configuration data about the hosts within that region. In particular, with this tool one could quickly check to see if too many hosts are going outside the region for time service instead of using a more appropriate local server.

Integration with Fremont

This NTP survey tool should be integrated with the Fremont discovery system [WCS93]. The NTP configuration program(s) could make use of the topology of the network that is recorded in the current Fremont database, and the NTP survey occasionally discovers hosts with multiple network interfaces that would be of interest to Fremont.

Acknowledgements

We would like to thank Dave Mills for helpful suggestions and comments about this work.

This work was supported in part by the National Science Foundation under grant numbers NCR-9105372 and NCR-9204853, the Advanced Research Projects Agency under contract number DABT63-93-C-0052, and an equipment grant from Sun Microsystems' Collaborative Research Program.

The information contained in this paper does not necessarily reflect the position or the policy of the U.S. Government or other sponsors of this research. No official endorsement should be inferred.

References

[BRA80] Braun, W.B., Short term frequency effects in networks of coupled oscillators. IEEE Trans. Communications COM-28,8 (August 1980), pp. 1269-1275.

[CQ92] S. Carl-Mitchell and J. S. Quarterman. Internet Firewalls. UNIX/World, 9(2), pp. 93-102, Tech Valley Publishing, Mountain View, California, February 1992.

[ED92] A. Emtage and P. Deutsch. Archie - An Electronic Directory Service for the Internet. Proceedings of the USENIX Winter Conference, pp. 93-110, San Francisco, California, January 1992.

[GAN92] Ganatra, N., CENSUS: Collecting Host Information on a Wide Area Network, University of Santa Cruz, technical report UCSC-CRL-92-34, Anonymous FTP from ftp.cse.ucsc.edu.

[HOB93] Hobbs, W.T., RDB Software Distribution. Available by anonymous FTP from rand.org, pub/RDB-hobbs/RDB-2.5j.tar.Z.

[LIN80] Lindsay, W.C., and A.V. Kantak, Network synchronization of random signals. IEEE Trans. Communications COM-28,8 (August 1980), pp. 1260-1266.

[MIL92] Mills, D.L., Network Time Protocol (Version 3) Specification, Implementation and Analysis. ARPA Network Working Group Report RFC-1305, University of Delaware, March 1992.

[MIL93a] Mills, D.L., Electronic mail of July 21, 1993 to the NTP electronic mailing list.

[MIL93b] Mills, D.L., Private communication on December 22, 1993.

[MOC93] Mockapetris, P.V., Private communication, 1993.

[PRUY93] Pruy, R., Electronic mail of October 28, 1993 to the NTP electronic mailing list.

[SNMP90] Schoffstall, M., Fedor, M., Davin, J., Case, J., A Simple Network Management Protocol (SNMP). ARPA Network Working Group Report RFC-1157, May 1990.

[SNS88] J. G. Steiner, B. C. Neuman and J. I. Schiller. Kerberos: An Authentication Service for Open Network Systems. Proceedings of the USENIX Winter Conference, pp. 191-201, February 1988.

[WCS93] Wood, D.C.M., Coleman S., and M. Schwartz, Fremont: A System for Discovering Network Characteristics and Problems. Proc. Winter USENIX Conference, pp. 78-89, January 1993.

[XNTP93] Mills, D.L., XNTP 3.3 Software Distribution. Available by anonymous FTP from louie.udel.edu, pub/ntp/xntp3.3.tar.Z.

[6] A simple heuristic might be to look for an NTP server running on routers and mail servers, and ignore systems that appeared to be Macs or PCs.

Author Information

James D. Guyton received his BA in Computer Science from the University of California at Santa Barbara in 1975. Since then he has worked at Xerox and the RAND Corporation. He has recently returned to graduate school and is working on an MS in Computer Science. Guyton can be reached by US Mail at the Computer Science Department, University of Colorado, Boulder, CO 80309-0430; or by electronic mail at guyton@cs.colorado.edu.

Michael F. Schwartz received his PhD in Computer Science from the University of Washington in 1987. He is currently an Assistant Professor of Computer Science at the University of Colorado, Boulder. His research focuses on issues raised by international networks and distributed systems, with particular focus on resource discovery and network measurement. Schwartz chairs an Internet Research Task Force Research Group on Resource Discovery and Directory Service, and is on the editorial boards for *IEEE/ACM Transactions on Networking* and for *Internet Society News*. Schwartz can be reached by US Mail at the Computer Science Department, University of Colorado, Boulder, CO 80309-0430; or by electronic mail at schwartz@cs.colorado.edu.